PTOL-413A (01-09)
Approved for use through 02/28/2009. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Applicant Initiated Interview Request Form

Application No.: 10/651,328 First Named Applicant: Amy H. Kang

Examiner: Wang, Rongfa Philip Art Unit: 2191 Status of Application: Final

Tentative Participants:

(1) Robert Kowert (2) Examiner Wang

(3) SPE? (4)

Proposed Date of Interview: July 1, 2009 **Proposed Time:** 3:00 PM Eastern AM/PM

Type of Interview Requested:

(1) ☑ Telephonic (2) ☐ Personal (3) ☐ Video Conference

Exhibit To Be Shown or Demonstrated: ☐ YES ☑ NO

If yes, provide brief description:

Issues To Be Discussed

Issues (Rej., Obj., etc)	Claims/ Fig. #s	Prior Art	Discussed	Agreed	Not Agreed
(1) Rejection	Indepen. Claims	Srivastava, H5E[6]	☐	☐	☐
(2)			☐	☐	☐
(3)			☐	☐	☐
(4)			☐	☐	☐

☑ Continuation Sheet Attached

Brief Description of Argument to be Presented:

See attached proposed amendment and agenda.

An interview was conducted on the above-identified application on _____ .

NOTE: This form should be completed by applicant and submitted to the examiner in advance of the interview (see MPEP § 713.01).

This application will not be delayed from issue because of applicant's failure to submit a written record of this interview. Therefore, applicant is advised to file a statement of the substance of this interview (37 CFR 1.133(b)) as soon as possible.

Applicant/Applicant's Representative Signature — Examiner/SPE Signature

Robert C. Kowert

Typed/Printed Name of Applicant or Representative

39,255

Registration Number, if applicable

1. (Currently amended) A system, comprising:

a processor; and

a memory comprising program instructions, wherein the program instructions are executable by the processor to implement:

in each of two or more threads of a multithreaded program, for each error generated by one or more functions executed in the thread, store an error trace element in ~~a memory storage area private to~~ thread private data of the respective thread, wherein the error trace element is stored by the thread to the respective thread private data in accordance with an application programming interface (API) to an error trace mechanism, and wherein the thread private data is a memory storage area of the multithreaded program accessible by the respective thread but not by others of the two or more threads; and

in a function of the multithreaded program, obtain an error trace for each of the two or more threads of the multithreaded program, wherein an error trace for a thread is obtained from the respective thread in accordance with the API to the error trace mechanism;

wherein each error trace includes one or more error trace elements ~~specific to the corresponding~~ from the thread private data of the respective thread, wherein each error trace element includes ~~information~~ a plurality of fields describing a particular error generated during execution of the ~~corresponding~~ respective thread.

Agenda:

1. The Office Action relies on Srivastava, paragraph [0003], to teach "two or more threads of a multithreaded program" and "store an error trace element in a memory storage area private to the thread [for each of the two or more threads]." The proposed amendment recites a thread storing an error trace element in thread private data of the respective thread.

As illustrated in Applicants' Fig. 2 and described in the accompanying description (e.g., paragraph [0021]) of Applicant's specification, each of the two or more threads in the multithreaded program is associated with a particular, separate and distinct, "thread private data", or "private data area", for the thread, which is exemplary of a memory storage area private to the corresponding thread. "Thread private data" is a term of art that refers to a memory storage area of a multithreaded program accessible by the respective thread but not by other threads. In contrast, paragraph [0003] of Srivastava states: "Often, each of the distributed nodes maintains a separate log file to store traces for their respective threads. Each distributed node may also maintain multiple trace logs corresponding to separate threads on that node." In addition, Srivastava's "trace logs" are described as logs of messages output by statements inserted in the application code by a programmer, as indicated in paragraph [0001]:

> Tracing is an approach for logging the state of computer applications at different points during its course of execution. Tracing is normally implemented by inserting statements in the computer application code that outputs status/state messages ("traces") as the statements are encountered during the execution of the code. Statements to generate traces are purposely placed in the computer application code to generate traces corresponding to activities of interest performed by specific sections of the code. The generated trace messages can be collected and stored during the execution of the application to form a trace log.

It is clear that what Srivastava is describing are trace log files maintained by and on the distributed nodes. This is further made clear in paragraph [0004], which begins "Diagnosing problems using multiple trace logs often involves a **manual** process of repeatedly inspecting different sets of the trace logs." Thus, Srivastava's "trace logs" are

clearly human-readable log files maintained on and by the distributed nodes. Srivastava's "trace logs" are clearly not thread private data as recited in Applicants' amended claim 1 and as understood as a term of art, but are instead trace log files maintained by the distributed nodes.

In addition, the H5E-A reference does not teach, for each error generated by one or more functions executed in a thread, the thread storing error trace elements to thread private data. [H5E-A does not disclose the use of thread private data].

2. The amended claim recites "in a function of the multithreaded program, obtain an error trace for each of the two or more threads of the multithreaded program, wherein an error trace for a thread is obtained from the respective thread in accordance with the API to the error trace mechanism, wherein each error trace includes one or more error trace elements from the thread private data of the respective thread, wherein each error trace element includes two or more fields describing a particular error generated during execution of the respective thread." No combination of the cited art teaches these limitations as recited in the amended claim.

3. There is no enabling description in the cited art for a multithreaded embodiment of the teachings of the H5E-A that would be achieved by such a combination, even if the combination were possible. The references do not describe how a multi-threaded version of the teachings of H5E-A would work, or in any way provide an enabling disclosure for a multi-threaded version of the teachings of H5E-A.

The Final Action states that "H5E-A shows at least a thread program… Srivastava discloses multiple threads…the combination appears to disclose multiple threads." However, "disclosing multiple threads" does not overcome Applicants' argument.

H5E-A refers to the document "The Error Handling Interface (H5E)." The Examiner has previously cited "H5E: Error Interface" (referred to as "H5E-B"). Page 1, paragraph 7, of H5E-B states (emphasis added):

> Each thread has its own error stack, but since multi-threading has not been added to the library yet, this package maintains a single error stack.

Thus, H5E-A/H5E-B acknowledges multi-threading. However, H5E-A/H5E-B as disclosed clearly does not support multi-threading. The mere fact that "Srivastava discloses multiple threads" does not provide an enabling disclosure for a multi-threaded version of the teachings of H5E-A. It is not clear how H5E-A could be modified to support multi-threading and the Office Action provides no explanation.